

A Hybrid AES-256 and Permissioned Blockchain Framework with Smart-Contract-Style Access Control for Tamper-Evident Educational Record Management: Design and Performance Evaluation

Suraj Kumar Sahu¹, Aradhana Kumari Singh²

Assistant Professor & PhD Research Scholar, Department of Computer Science & Engineering, MATS University,
Raipur, Chhattisgarh, India¹

Associate Professor, Department of Computer Science & Engineering, MATS University, Raipur, Chhattisgarh, India²

Abstract: Educational institutions hold academic records whose confidentiality and integrity are essential for credential verification. This paper presents a four-layer framework combining AES-256-CBC encryption, SHA-256 hash anchoring on a permissioned ledger, off-chain ciphertext storage, and smart-contract-style role-based access control with an append-only audit log. A Python prototype was evaluated functionally on three anonymised records and quantitatively on 200 synthetic records using a university marks-sheet schema, with metrics averaged over seven runs. Mean encryption and decryption times were 0.0354 ms and 0.0227 ms per record, while mean block-append latency was 0.0068 ms. Encrypted storage increased by 37.85%, whereas the on-chain digest footprint remained 64 bytes per record. All 20 deliberately corrupted records were detected. The single-node in-memory ledger makes throughput figures computational upper bounds rather than production measurements.

Keywords: student data security; AES-256; permissioned blockchain; SHA-256 hash anchoring; role-based access control; smart contracts; tamper detection; educational records

1. INTRODUCTION

Academic records such as marks, grades, transcripts and certificates are used for admissions, scholarships and employment decisions. They are also personal data, and institutions are expected to keep them confidential while still allowing third parties to confirm that a record is genuine. Centralised record systems satisfy neither requirement well: a privileged insider or an intruder can alter a stored value without leaving evidence that an outside party can check, and a single compromised server exposes every record.

Blockchain technology [24] offers tamper-evident, auditable record keeping and has been applied to education, mainly for credential and certificate verification [4], [6], [13], [25], [26]. However, writing personal data directly to a ledger is costly and conflicts with data-protection principles such as the right to erasure [16], [28]. Symmetric encryption, notably AES [1], protects data at rest efficiently, and hybrid models that encrypt first and anchor on a ledger have been proposed for medical records [7], [8]. What is still missing is a single, empirically evaluated design that joins encryption, hash anchoring, a permissioned l

edger and enforceable role-based access for the full range of student academic records [17].

This paper addresses that gap. Its contributions are:

- a four-layer architecture (data, security, storage/ledger, access control) that keeps only a fixed 64-byte SHA-256 digest on the ledger and stores AES-256 ciphertext off-chain;
- a smart-contract-style RBAC specification for four stakeholder roles (Admin, Faculty, Student, Verifier) with an ownership check and a complete audit trail, in which no role may delete a committed record;
- an executable prototype with a reproducible evaluation of encryption and decryption time, ledger latency, throughput, storage overhead and tamper detection, reported with run-to-run variation; and
- an explicit account of what the evaluation does and does not show, including the limits of a single-node simulation.

The remainder of the paper is organised as follows. Section 2 reviews related work, Section 3 describes the framework, Section 4 the experimental setup, Section 5 the results, Section 6 discusses them, Section 7 states limitations and threats to validity, and Section 8 concludes.

2. RELATED WORK

Encryption for academic data. AES, standardised by NIST in 2001 [1], is the benchmark symmetric cipher, and its use for institutional data has been shown to be practical at acceptable computational cost [2], [3]. These works, however, rely on centralised storage and offer no immutable record of changes.

Blockchain in education. Sharples and Domingue proposed a blockchain system for educational records, reputation and rewards [4]; EduChain addressed education data management more broadly [5]; Cerberus targeted accreditation and degree verification [6]; and EduCTX proposed a blockchain-based higher-education credit platform [25]. A European Commission report surveyed the opportunities and risks of blockchain in education [26], and Kshetri discussed the role of blockchain in cybersecurity and privacy [29]. Systematic and bibliometric reviews conclude that most initiatives remain focused on certificate verification and call for integrated, multi-layered security architectures [17].

Hybrid AES–blockchain models. Yuan and Wang combined AES with a blockchain for medical records [7], and Hidayat and Mahardiko reviewed such integrations, concluding that the combination strengthens both confidentiality and integrity [8]. Mahajan and Sachdeva applied AES to academic certificate validation [9], but their scope is limited to certificates.

Access control and advanced cryptography. Role-based access control [21], [22] is a standard authorisation model, and RBAC-SC implements it through smart contracts [10]. Zyskind et al. proposed a blockchain-based access-control layer for personal data [23]. Attribute-based encryption [12] and zero-knowledge-proof schemes for student records [11] offer fine-grained privacy but add considerable computational overhead. Off-chain storage with on-chain hashes is a recognised scalability pattern [14], [15], [27], and permissioned Hyperledger-based designs have been applied to certificate privacy [13]. GDPR-oriented studies highlight the conflict between ledger immutability and erasure rights [16].

Table 1 summarises the studies closest to this work and the limitation each leaves open.

Table 1. Closest related studies and the limitation addressed in this work

Study	Technique	Domain	Limitation addressed here
Yuan & Wang [7]	AES + blockchain	Healthcare records	Domain-specific; not adapted to education access hierarchies and record formats
Mahajan & Sachdeva [9]	Blockchain + AES	Academic certificates	Limited to certificate validation, not the wider record lifecycle
Sharples & Domingue [4]	Blockchain credentials	Educational certificates	Focus on credential verification; no encryption layer for stored records
Li, Tan & Tian [11]	ZKP + proxy re-encryption	Student academic records	High computational cost for large-scale deployment
Cruz et al. [10]	Smart-contract RBAC	Generic blockchain systems	Not tailored to student, faculty and verifier roles
DCVPC and related [13]	Hyperledger Fabric	Academic certificates	Limited encryption depth for stored data fields
Nizamuddin et al. [14]	IPFS off-chain storage	Blockchain data storage	Off-chain data not always cryptographically linked to the on-chain hash

To the best of our knowledge from this review, few studies unify AES-256 encryption, SHA-256 anchoring, a permissioned ledger and smart-contract-style RBAC for student academic records and evaluate the combination empirically. The framework below is designed to do so.

3. PROPOSED FRAMEWORK

3.1 Architecture

The framework has four cooperating layers (Fig. 1). The Data Layer structures each record as a key-value object (student identifier, course, batch, semester, subject-wise credit, grade and grade point, credits earned, grand total, SGPA and result). The Security Layer encrypts the record and hashes the ciphertext. The Storage/Ledger Layer keeps the ciphertext off-chain and anchors only its hash on a permissioned chain. The Access Control Layer decides who may perform which action and logs every decision.

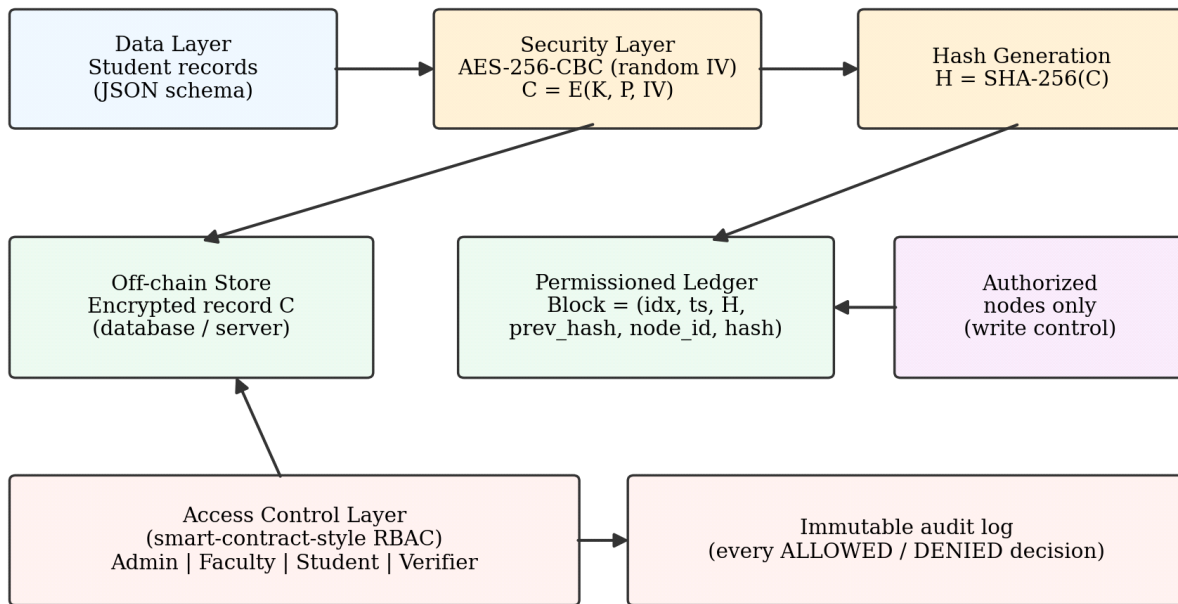


Fig. 1. Four-layer hybrid AES-256 and permissioned blockchain architecture.

3.2 Record protection

Each record P is serialised to JSON and encrypted with AES-256 in CBC mode [18] under key K , using a fresh random 128-bit initialisation vector IV for every operation so that identical records never yield identical ciphertext. The IV is prepended to the ciphertext and the result is Base64-encoded for storage:

$$C = E(K, P, IV) \quad (1)$$

$$P = D(K, C, IV) \quad (2)$$

The integrity fingerprint of the stored ciphertext is its SHA-256 digest [19]:

$$H = \text{SHA-256}(C) \quad (3)$$

3.3 Permissioned ledger

Only pre-registered institutional nodes (for example, a registrar office) may append blocks, as in permissioned platforms such as Hyperledger Fabric [20]. This design reflects the institutional trust model of a university and avoids anonymous proof-of-work consensus. Each block holds an index, a timestamp, the record hash H , the previous block's hash, the identifier of the submitting node, and its own block hash:

$$\text{Hash}_i = \text{SHA-256}(i \parallel t_i \parallel H_i \parallel \text{Hash}_{(i-1)} \parallel \text{node}_i) \quad (4)$$

Altering any block changes its hash and breaks the previous-hash link stored in every later block, so a sequential recomputation both detects tampering and localises it to the first broken link. Because only H is stored on the ledger, the ledger grows with the number of records and not with their size.

3.4 Smart-contract-style RBAC

Access decisions are taken from a fixed permission matrix (Table 2). The VIEW_OWN_RECORD action adds an ownership condition: a Student is allowed only when the requesting identifier equals the record owner. Every decision, allowed or denied, is appended to an audit log with timestamp, user, role, action and outcome.

Table 2. RBAC permission matrix

Action	Authorised roles	Rationale
ADD_RECORD	Admin, Faculty	Only academic staff originate records
VIEW_OWN_RECORD	Student (own record only)	Protects other students' data
VIEW_ANY_RECORD	Admin, Faculty	Administrative oversight
VERIFY_RECORD	Verifier, Admin	Third-party verification without edit rights
UPDATE_RECORD	Admin	Corrections restricted to the highest-trust role
DELETE_RECORD	None	Enforces append-only, immutable behaviour

3.5 Record verification

To verify a record, a Verifier (after an RBAC check) proceeds as follows:

1. retrieve the stored ciphertext C' and its block index i from the off-chain store;
2. recompute $H' = \text{SHA-256}(C')$;
3. compare H' with the record hash stored in block i ; a mismatch indicates modification after anchoring; and
4. optionally validate the whole chain by recomputing every block hash and previous-hash link.

Verification needs neither the plaintext nor the AES key, so a third party can confirm integrity without gaining access to the content.

3.6 Erasure by key destruction

Since the ledger holds only digests, destroying the AES key renders the off-chain ciphertext unreadable without altering the immutable ledger. This is a design property motivated by the erasure concerns in [16]; it was not experimentally evaluated in this study.

4. EXPERIMENTAL SETUP

The framework was implemented in Python 3 using PyCryptodome for AES-256-CBC, the standard hashlib module for SHA-256, a custom permissioned-chain class, and a custom RBAC class. Timing used the Python `time.perf_counter` function and the statistics module. All experiments ran on a single machine ([INSERT ACTUAL CPU MODEL, RAM, OPERATING SYSTEM, AND PYTHON VERSION USED FOR THE EXPERIMENTS]) without network transmission.

Datasets. Functional testing used three anonymised sample records (semesters 7, 7 and 3; results PASS and ATKT) whose structure mirrors a real university marks sheet. Performance testing used $N = 200$ synthetic records with the same schema (six subject entries each, SGPA, credits earned, grand total, result), giving a mean plaintext size of 713.8 bytes. Student names and enrolment numbers were anonymised because the source documents contain personal data of third parties.

Metrics and protocol. We measured (i) per-record AES-256 encryption and decryption time; (ii) per-block append latency, total time and throughput for 200 blocks; (iii) storage overhead of AES plus Base64 relative to plaintext and the on-chain bytes per record; and (iv) tamper detection. For tamper detection, 20 of the 200 ciphertexts had their final four characters replaced and were re-verified against their on-chain hashes. Every metric was collected over seven independent runs; we report a representative single run and the seven-run average because single-run timings on a shared machine vary considerably.

5. RESULTS

5.1 Functional correctness

Table 3 lists the functional scenarios executed with the end-to-end prototype. All produced the expected outcome.

Table 3. Functional test scenarios and outcomes

Scenario	Expected	Observed
Encrypt and anchor three records	Blocks #1–#3 created; chain valid	As expected; chain reported valid
Student requests own record	ALLOWED	ALLOWED; record decrypted
Student requests another student's record	DENIED	DENIED
Verifier verifies a stored record	ALLOWED; hash matches	ALLOWED; VALID
Off-chain ciphertext corrupted, then verified	Mismatch reported	TAMPERED detected
Audit log after the session	Every decision logged	12 events recorded

5.2 Encryption and decryption time

Table 4 shows that both operations take well under a millisecond per record. The seven-run averages are lower than the representative single run because the single run includes a 3.2166 ms outlier in encryption, which illustrates why averaged values are more reliable.

Table 4. AES-256 encryption and decryption time (N = 200 records)

Metric	Single run	7-run average
Encryption, mean	0.0528 ms	0.0354 ms
Encryption, min / max	0.0299 / 3.2166 ms	—
Decryption, mean	0.0352 ms	0.0227 ms
Decryption, min / max	0.0295 / 0.0922 ms	—

5.3 Ledger latency and throughput

Appending a block costs only hashing and pointer chaining, so its mean latency (0.0068 ms, seven-run average) is lower than that of AES operations (Table 5, Fig. 2). Throughput was 77,467 transactions/s in the representative run and about 144,548 transactions/s as the seven-run average. These values describe an in-memory single-node simulation and must be read as computational upper bounds (Section 7).

Table 5. Block-append latency and throughput (N = 200 blocks)

Metric	Single run	7-run average
Block-add latency, mean	0.0127 ms	0.0068 ms
Block-add latency, min / max	0.0105 / 0.0522 ms	—
Total time for 200 blocks	2.58 ms	≈ 1.4–1.6 ms
Throughput	77,467 tx/s	≈ 144,548 tx/s

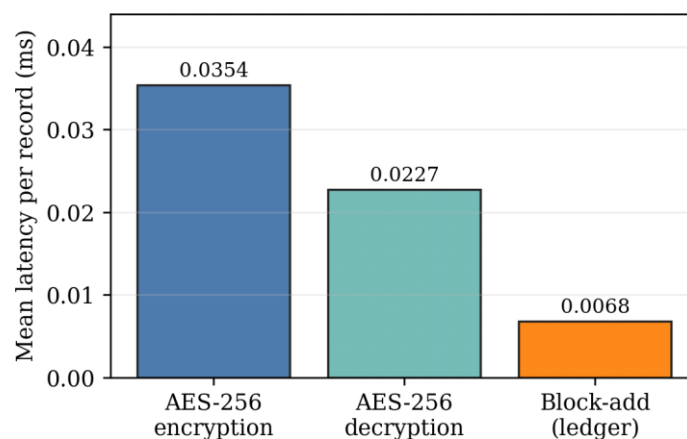


Fig. 2. Mean latency per operation (N = 200, seven-run average).

5.4 Storage overhead and on-chain footprint

Encrypted records stored off-chain were 37.85% larger than plaintext (984.0 versus 713.8 bytes on average), whereas the on-chain footprint was a constant 64 bytes per record irrespective of record size (Table 6, Fig. 3).

Table 6. Storage per record

Data form	Average size (bytes)	Overhead vs. plaintext
Plaintext record	713.8	—
AES-256 ciphertext, Base64 (off-chain)	984.0	+37.85%
SHA-256 digest (on-chain)	64	Fixed, size-independent

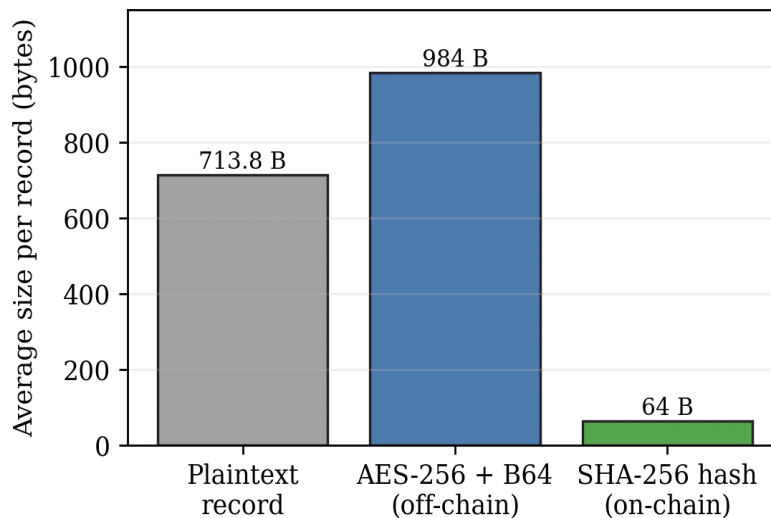


Fig. 3. Average storage footprint per record.

5.5 Tamper detection and chain integrity

All 20 corrupted ciphertexts were flagged as invalid (detection rate 20/20 = 100%). A full validity check over all 200 blocks also succeeded, confirming that the previous-hash linkage remained internally consistent.

6. DISCUSSION

Cost of confidentiality. At roughly 0.035 ms to encrypt and 0.023 ms to decrypt a record of about 700 bytes, AES-256 adds a negligible delay to record handling, so strong encryption does not have to be traded against responsiveness. This supports the choice of a lightweight symmetric cipher over heavier mechanisms such as ABE or zero-knowledge proofs for this workload [11], [12], although those schemes provide privacy properties that AES alone does not.

Origin of the storage overhead. The 37.85% increase is largely structural. A 713.8-byte plaintext is padded to the next 16-byte boundary, a 16-byte IV is prepended, and Base64 then expands the result by a factor of 4/3, which together give approximately the observed 984 bytes. Storing raw ciphertext instead of Base64 would remove most of the overhead, at the cost of a binary-safe store.

Scalability of the ledger. Only a 64-byte digest is stored per record, so ledger growth depends on the number of records and not their size. At 64 bytes per record, one million records would add about 64 MB of digests before block metadata, a figure that follows arithmetically from the design rather than from measurement. The reported throughput excludes network latency, endorsement and consensus, and a production network would be materially slower.

Security properties. Table 7 relates each security goal to the mechanism that provides it and the assumption on which it depends.

Table 7. Security goals, mechanisms and assumptions

Goal	Mechanism	Assumption / caveat
Confidentiality	AES-256-CBC with random IV	Key K is kept secret; key management is not evaluated
Integrity of stored records	SHA-256 digest anchored on ledger	Digest on the ledger is itself protected from rewriting
Ledger tamper evidence	Hash chaining of blocks	Replication across independent nodes is needed against a malicious writer
Access control	Permission matrix with ownership check	Caller identity is assumed authentic; authentication is outside the prototype
Accountability	Append-only audit log	Log storage is protected as part of the ledger

AES-CBC does not authenticate ciphertext, so modification of stored data is detected by the on-chain hash comparison rather than by the cipher itself; an authenticated mode such as AES-GCM would add a second, independent check. The 100% detection result is expected from the collision resistance of SHA-256 for any altered ciphertext and mainly confirms that the implementation is correct.

7. LIMITATIONS AND THREATS TO VALIDITY

- **Simulated ledger.** The chain is an in-memory structure on one machine, with no peer network, consensus protocol or replication. Tamper evidence against a malicious authorised writer who rewrites the whole chain is therefore not demonstrated.
- **Throughput is an upper bound.** Reported transactions per second exclude network, endorsement, consensus and disk I/O and are not comparable with figures from platforms such as Hyperledger Fabric.
- **Synthetic and small data.** Performance used 200 synthetic records of a single schema, and the functional test used three records. Larger documents, scanned files and concurrent multi-user load were not tested.
- **Smart contract not deployed.** RBAC is a Python class that mirrors chaincode or Solidity logic; it has not been deployed on a real platform, and the demonstrator identifies users without authentication.
- **Key management.** A single AES key is held in memory; rotation, storage in an HSM and per-record keys were not evaluated.
- **Measurement noise.** Microsecond-scale timings on a general-purpose machine vary between runs (for example, a 3.2 ms encryption outlier), which is why seven-run averages are reported.
- **Narrow tamper test.** Tamper detection covered corruption of off-chain ciphertext (20 samples), not block-level forgery or key compromise.

8. CONCLUSION AND FUTURE WORK

This paper presented a hybrid framework that protects student academic records with AES-256 encryption, anchors only SHA-256 digests on a permissioned ledger, and governs access through a smart-contract-style RBAC layer with a full audit trail. The prototype behaved correctly in all functional scenarios, encrypted and decrypted records in about 0.035 ms and 0.023 ms, appended blocks in about 0.007 ms, kept the on-chain footprint at a constant 64 bytes per record, and detected all 20 tampered records. These results show that the design is computationally lightweight and tamper-evident within the scope of a single-node simulation.

Future work will deploy the ledger and access-control logic as chaincode on a multi-node Hyperledger Fabric network [20] and measure latency and throughput under consensus; compare AES-CBC with authenticated AES-GCM; evaluate key management and rotation; test with larger records and concurrent users on real institutional data under appropriate consent; and assess crypto-erasure against regulatory requirements.

DECLARATIONS

Data and code availability. The prototype and benchmark scripts are not yet publicly archived. The study uses anonymized or synthetic records only.

Ethics. No real student identities are included; identifiers in the sample records were replaced with generic labels.

Conflict of interest and funding. The authors declare that no external funding was received for this study and no conflict of interest is declared.

REFERENCES

- [1] J. Daemen and V. Rijmen, *The Design of Rijndael: AES – The Advanced Encryption Standard*. Berlin, Germany: Springer, 2002; see also NIST, FIPS PUB 197: *Advanced Encryption Standard (AES)*, 2001.
- [2] N. Ferguson and B. Schneier, *Practical Cryptography*. Indianapolis, IN, USA: Wiley, 2003.
- [3] Waybhase and Adakane, *AES-based security of academic and institutional data*, 2022.
- [4] M. Sharples and J. Domingue, “The blockchain and kudos: a distributed system for educational record, reputation and reward,” in *Proc. EC-TEL 2016, LNCS vol. 9891*. Springer, 2016, pp. 490–496.
- [5] Liu et al., *EduChain: a blockchain-based education data management system*.
- [6] Tariq, Haq and Ali, *Cerberus: a blockchain-based accreditation and degree verification system*.
- [7] Yuan and Wang, *blockchain system with AES encryption for medical records*, 2018.
- [8] Hidayat and Mahardiko, *review of AES–blockchain integration strategies*.
- [9] Mahajan and Sachdeva, *blockchain-based framework for validation and authentication of academic certificates using AES*, 2022.
- [10] J. P. Cruz, Y. Kaji and N. Yanai, “RBAC-SC: Role-based access control using smart contract,” *IEEE Access*, vol. 6, pp. 12240–12251, 2018.
- [11] Li, Tan and Tian, *secure and efficient sharing framework for student electronic academic records using zero-knowledge proofs and proxy re-encryption*, 2026.
- [12] V. Goyal, O. Pandey, A. Sahai and B. Waters, “Attribute-based encryption for fine-grained access control of encrypted data,” in *Proc. 13th ACM Conf. Computer and Communications Security (CCS)*, 2006, pp. 89–98.
- [13] DCVPC: *privacy-preserving protocol for academic certificates based on Hyperledger Fabric, and related Hyperledger-based studies*, 2023–2025.
- [14] Nizamuddin et al., *IPFS-blockchain based off-chain storage for blockchain applications*.
- [15] Jayabalan and Jeyanthi, *scalable blockchain model using off-chain IPFS storage for healthcare data security and privacy*.
- [16] *Studies on GDPR compliance and the right to be forgotten in blockchain systems*, 2018–2022.
- [17] *Systematic and bibliometric reviews of blockchain in education and blockchain-driven academic learning record management*, 2024–2025.
- [18] M. Dworkin, *Recommendation for Block Cipher Modes of Operation: Methods and Techniques*, NIST Special Publication 800-38A, 2001.
- [19] NIST, FIPS PUB 180-4: *Secure Hash Standard (SHS)*, 2015.
- [20] E. Androulaki et al., “Hyperledger Fabric: A distributed operating system for permissioned blockchains,” in *Proc. 13th EuroSys Conf.*, 2018, Art. 30.
- [21] D. F. Ferraiolo and D. R. Kuhn, “Role-based access controls,” in *Proc. 15th NIST-NCSC National Computer Security Conf.*, 1992, pp. 554–563.
- [22] R. S. Sandhu, E. J. Coyne, H. L. Feinstein and C. E. Youman, “Role-based access control models,” *IEEE Computer*, vol. 29, no. 2, pp. 38–47, 1996.
- [23] G. Zyskind, O. Nathan and A. Pentland, “Decentralizing privacy: Using blockchain to protect personal data,” in *Proc. IEEE Security and Privacy Workshops (SPW)*, 2015, pp. 180–184.
- [24] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” 2008.
- [25] M. Turkanović, M. Hölbl, K. Košič, M. Heričko and A. Kamišalić, “EduCTX: A blockchain-based higher education credit platform,” *IEEE Access*, vol. 6, pp. 5112–5127, 2018.
- [26] A. Grech and A. F. Camilleri, *Blockchain in Education*, JRC Science for Policy Report EUR 28778 EN, Publications Office of the European Union, 2017.
- [27] J. Benet, “IPFS – Content addressed, versioned, P2P file system,” *arXiv:1407.3561*, 2014.
- [28] European Parliament and Council, *Regulation (EU) 2016/679 (General Data Protection Regulation)*, 2016.
- [29] N. Kshetri, “Blockchain’s roles in strengthening cybersecurity and protecting privacy,” *Telecommunications Policy*, vol. 41, no. 10, pp. 1027–1038, 2017.